



# Getting Cozy with Java HotSwap

The Java HotSwap technology saves precious hours for development teams by cutting down on the time required to fix bugs. Interested? We bring you the basics!

**T**here may be times when you are required to find bugs in a running application, fix them and test the fix without restarting the application all over again. This could be a common requirement if you are working on enterprise level applications, where the acceptable downtime is short. In this article, we shall discover how to use the Java HotSwap technology to tackle such situations with ease.

The Java HotSwap technology is based on the Java Platform Debugger Architecture (JPDA). Figure 2 shows a high-level

architecture view of the Java HotSwap technology.

The HotSwap client tool implements the Java Debugger Interface (JDI). There are lots of free client tools available in the market, which support the HotSwap functionality. IBM's Eclipse and Sun's HotSwap Client Tool are just a few of them.

The HotSwap client tool communicates with a local or remote HotSwap capable JVM using the Java Debugger Wire protocol. The power and usefulness of the Java HotSwap technology may be best explained with the help of a simple example.

## HOTSWAP TECHNOLOGY—THE BASICS

HotSwap adds a feature to the Java platform (starting with Java 1.4) that allows the replacement of Java classes on the fly. This dynamic class redefinition helps the developer to replace old class instances with new ones. And, what's more, this does not require restarting the running application. Figure 1 illustrates the advantages of the Java HotSwap technology.

If you want to test your fix without making use of the HotSwap technology, you would need to restart the running application after applying your fix. This, definitely, is a time consuming and annoying process. But with the HotSwap technology at your disposal, you don't need to restart your application after applying the fix.

Consider the Java program listed in Figure 3.

This program is so simple that it just displays the message 'power of java hotswap technology' in an infinite 'while' loop having a sleep interval of 1000 ms.

We shall modify this program on-the-fly using the Java HotSwap technology so that it displays the string in the following format: **The Power of the Java HotSwap Technology**. See the magic unfold as you run the program. We shall use Sun's HotSwap client tool for dynamic class redefinition. See the References section for download URLs.

Just start the original program in the debug mode by using the following commands.

```
java -hotspot -
Xrunjdwp:transport=dt_socket,server=y,suspend=n,
address=4000 -Xdebug -Xnoagent -cp . Simple
```

Now, launch Sun's HotSwap client, and configure the old and new class path and source path locations. Next, configure the connection settings as shown in Figure 4.

Now, connect Sun's HotSwap client to your application by selecting the *JVM→Connect* menu option. When successfully connected, you should see a pop-up window similar to what is shown in Figure 5.

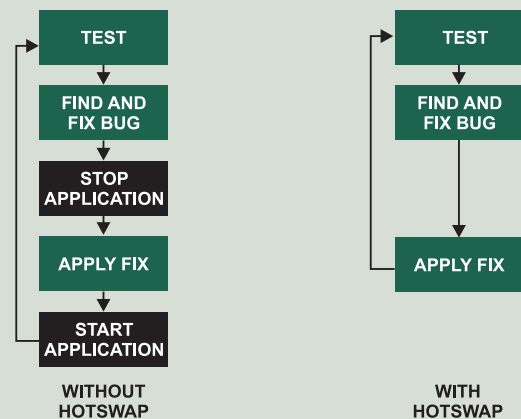
The pop-up window describes the remote JVM capability for dynamic class redefinition. Once connected, you can redefine the old class by loading the modified class and selecting either the *File→Open New Class File* or the *JVM→Find Changed Classes* menu option.

Modify and compile our simple program to display the message in the format—**The Power of the Java Hotspot Technology**.

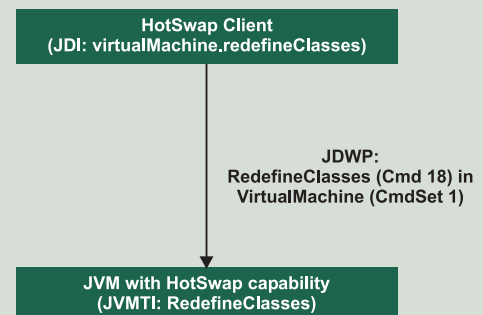
Selecting *JVM→Find Changed Classes* in the Sun's HotSwap client tool will display a list of the modified classes as shown in Figure 6.

From this figure, you can see that the only difference between our old and new program is the string format. To submit the modified class, click on the *Submit* button. Once submitted, you can see that the changes we made to

## FIGURE 1: ADVANTAGES OF THE JAVA HOTSWAP TECHNOLOGY



## FIGURE 2: THE ARCHITECTURE OF THE JAVA HOTSWAP TECHNOLOGY



```
public class Simple
{
    public void display()
    {
        System.out.println("power of java hotswap technology!!!");
    }

    public static void main(String args[])
    {
        Simple s = new Simple();

        while(true) {
            s.display();

            try {
                Thread.sleep(1000);
            } catch (Throwable ignore) {
            }
        }
    }
}
```

Figure 3: A simple Java program illustrating the HotSwap technology



Figure 4: Sun's HotSwap client—the connection settings window



Figure 5: Sun's HotSwap client—the connection status

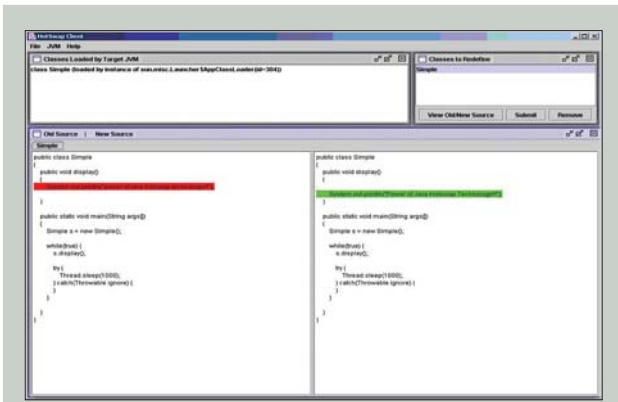


Figure 6: Sun's HotSwap client tool

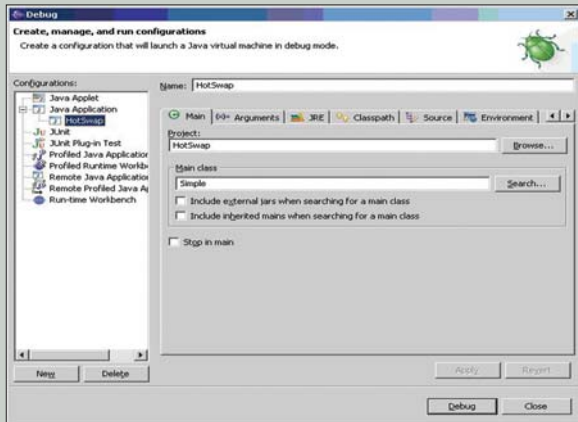


Figure 7: The Eclipse debug launch window

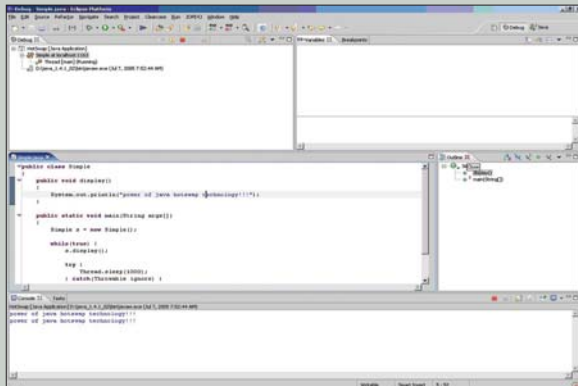


Figure 8: The Eclipse debug perspective

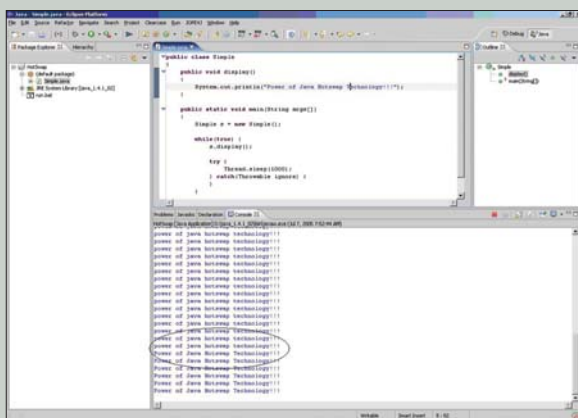


Figure 9: Automatic HotSwapping with Eclipse

## COUNTING THE SAVINGS

The HotSwap technology puts the following benefits in the hands of developers:

- It facilitates on-the-fly patching for applications that cannot be interrupted.
- It facilitates on-the-fly debugging by allowing the developer to add more print statements to a running application.
- It facilitates profiling (NetBeans' JFluid uses the Code HotSwapping technology).


the modified class will get reflected in our running application. This is shown below.

### Listing 1: Dynamic class redefinition

```
java -hotspot -Xrunjdw:trans
port=dt socket,server=y,suspend=n,address=4000 -Xdebug
-Xnoagent -cp.Simple
power of java hotswap technology!!!
power of java hotswap technology!!!
power of java hotswap technology!!!
Power of java Hotswap Technology!!!
Power of java Hotswap Technology!!!
Power of java Hotswap Technology!!!
Power of java Hotswap Technology!!!
```

The following steps show how you can achieve the same thing using the much-discussed Eclipse tool.

- Create an Eclipse project for your simple program.
- Build and run the project in the debug mode as shown in Figures 7 and 8.
- While running in the debug mode, change the string format and compile the project for Eclipse to automatically do a HotSwap for you as shown in Figure 9. Just remember that we have done all this without restarting the already running application. Really great, isn't it?

The Java HotSwap technology saves precious hours for development teams by considerably cutting down on the time required to fix bugs. In an enterprise-level application development environment, this tool could prove to be particularly useful. Having covered the basics required to get you started with the tool, you will surely want to explore further possibilities with the technology. **END** 

## REFERENCES

- <http://java.sun.com>
- <http://java.sun.com/j2se/1.4.2/docs/guide/jpda/>
- <http://www.experimentalstuff.com/Technologies/HotSwapTool/index.html>

**By: Raja R K.** The author is a lead engineer with HCL Technologies, Chennai.